**WEEK1 REPORT**

**Greedy or Not**

For each subarray of the given list, we can associate with it a number which represents the difference between points of the first player and second player when both the players play optimally (let’s call it max\_diff(array) ).

For a subarray of length 1 it is just the value of the only existing element.

For subarrays of length > 1 (let’s call the subarray a1,a2,.....,an-1,an), the value associated

(max\_diff( a1,a2,.....,an-1,an)) =

maximum(a1 - max\_diff(a2 ,….an-1, an) , an - max\_diff(a1,a2,.....an-1))

The first player has two options, either he chooses a1 or an. After that the second player will have the move and that is the reason for the negative sign. He will obviously go for one which has greater value and so we take the maximum.

Dynamic Programming solution has been implemented both in python and c++.

**Nim**

1. No, there doesn't always exist a sequence of moves such that the player making the first move wins. Simplest example is the sequence 2,1 which is losing for the player moving first.
2. First brute-force approach was used to check for outcomes in various states. It was difficult for me to find a pattern but then with help i figured out that if the xor of matchsticks in all piles is zero only then the position would be lost for the first player.
3. I tried to prove this myself and implemented a strategy that would play optimally.

**Pawnscape**

1. White on the first move should play one of the corner pawns. This ensures that with perfect play white doesn't lose. If white plays one of the middle pawns, they would lose even with perfect play.
2. With perfect play from both sides it is a draw.
3. …
4. There does not exist a strategy which ensures win for black with perfect play for both players.
5. There does exist a strategy for n = 4 such that black never loses

For n = 5 white wins if they play the middle pawn first.

**CHOMP**

1. I couldn’t come up with a proof though I couldn't find a counterexample either.
2. Just like nim we can use brute-force approach to try out all the possible moves for a player. If for the player playing the first move there exists even a single position which he/she can reach which is losing for the second player (who will have the move the next time) then he/she can play that move and win . If there is only a single row then the position is lost for the player playing first iff there is only 1 chocolate piece. For other positions we could try all the combinations and check if it is possible to reach a position which would be losing for the second player.